

CSE-3505

Foundations of Data Analytics

J Component Report

A project report titled

STOCK VALUE PREDICTION

By

19BEC1327 PRASSANTH A

19BEC1330 VENKATSUNDHAR SP

19BEC1345 LOKESH A

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Submitted to

Dr. R. KARTHIK

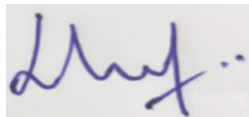
NOV 2021

School of Electronics Engineering

DECLARATION BY THE CANDIDATE

I hereby declare that the Report entitled “**STOCK MARKET PREDICTION**” submitted by me to VIT Chennai is a record of bonafide work undertaken by me under the supervision of **Dr. R. Karthik, Senior Assistant Professor, SENSE, VIT Chennai.**

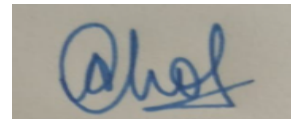
VenkatSundhar.SP



Prassanth.A



Lokesh.A



Chennai

10/12/2021

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. R. Karthik**, School of Electronics Engineering for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr. Sivasubramanian. A**, Dean of the School of Electronics Engineering (SENSE), VIT University Chennai, for extending the facilities of the School towards our project and for his unstinting support.

We express our thanks to our **Head of The Department Dr. Vetrivelan. P** (for **B.Tech-ECE**) for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the courses till date.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

BONAFIDE CERTIFICATE

Certified that this project report entitled “**STOCK MARKET PREDICTION**” is a bonafide work of **PRASSANTH A (19BEC1327)** , **VENKATSUNDHAR SP (19BEC1330)** and **LOKESH A (19BEC1345)** carried out the “J”-Project work under my supervision and guidance for CSE 3505 Foundation of Data Analytics.

Dr.R.Karthik

School of Electronics Engineering

VIT University, Chennai

Chennai – 600 127.

TABLE OF CONTENTS

S.NO	SUB	CHAPTER	PG.NO
1		Chapter -1 Introduction	7
	1.1	Objective	7
	1.2	Key Terms in Stock Market	7
2		Chapter – 2 Requirements and proposed system	8
	2.1	Software Used	8
	2.2	Datasets	8
3		Chapter -3 Module description	10
	3.1	Numerical Analysis	10
	3.2	Textual Analysis	11
4		Chapter 4 – Results and Discussion	14
	4.1	Numerical Analysis	14
	4.2	Textual Analysis	19
5		Chapter 5 - Conclusion	23
6		Reference	24
7		Appendix	25

ABSTRACT

The act of attempting to anticipate the future value of a business stock or other financial instrument traded on an exchange is known as stock market prediction. The successful prediction of a stock's future price could yield significant profit. Stock prices, according to the efficient-market theory, represent all currently accessible information, and any price changes that are not based on newly revealed information are thus fundamentally unpredictable.

Information and rational expectations drive stock prices, and newly revealed information about a company's prospects is almost immediately reflected in the present stock price. This would indicate that all publicly available information about a company, including its price history, is already reflected in the stock's current price. Accordingly, changes in the stock price reflect release of new information, changes in the market generally, or random movements around the value that reflects the existing information set.

CHAPTER-1

INTRODUCTION

1.1 Objective

In the past decades, there is an increasing interest in predicting markets among economists, policy makers and academics. The objective of the proposed work is to implement various analysis like numerical and textual analysis to predict the stock market.

1.2 Key terms in stock market:

- **Open value**

The opening price is the price at which a security first trades upon the opening of an exchange on a trading day; for example, the New York Stock Exchange (NYSE) opens at precisely 9:30 a.m. Eastern time. The price of the first trade for any listed stock is its daily opening price. The opening price is an important marker for that day's trading activity, particularly for those interested in measuring short-term results such as day traders.

- **Close value**

The closing price is the raw price or cash value of the last transacted price in a security before the market officially closes for normal trading. It is often the reference point used by investors to compare a stock's performance since the previous day—and closing prices are frequently used to construct line graphs depicting historical price changes over time.

The adjusted closing price factors in anything that might affect the stock price after the market closes, such as dividends or splits. Most stocks and other financial instruments are traded after-hours, although in far smaller volumes. Therefore, the closing price of any security is often different from its after-hours trading price.

- **Adjusted value:** The adjusted closing price amends a stock's closing price to reflect that stock's value after accounting for any corporate actions. It is often used when examining historical returns or doing a detailed analysis of past performance.

CHAPTER-2

REQUIREMENTS AND PROPOSED SYSTEM

2.1 Software Used

Jupyter Notebook: Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating notebook documents.

A Jupyter Notebook document is a browser-based REPL containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media. Underneath the interface, a notebook is a JSON document, following a versioned schema, usually ending with the ".ipynb" extension.

Project Jupyter's name is a reference to the three core programming languages supported by Jupyter, which are Julia, Python and R.

2.2 Datasets

1)Numeric:

Yahoo! Finance [Historical Stock Data]: Yahoo! Finance is a media property that is important for Yahoo! network. It gives monetary news, information and editorial including stock statements, official statements, monetary reports, and unique substance. It additionally offers some internet based instruments for individual accounting of the executives. As well as posting accomplice content from other sites, it posts unique stories by its group of staff columnists. It is positioned twentieth by SimilarWeb on the rundown of biggest news and media sites.

2)Textual:

Harvard Dataverse [Times Of India Headlines]: The Dataverse is an open source web application to share, safeguard, refer to, investigate and break down research information. Specialists, information creators, distributors, information wholesalers, and associated establishments all get suitable credit by means of an information reference with a persevering identifier (e.g., DOI, or Handle).

A Dataverse storehouse has different dataverses. Each dataverse contains dataset(s) or different information refrains, and each dataset contains enlightening metadata and information records (counting documentation and code that go with the information).

A coordinated effort with the Institute for Quantitative Social Science (IQSS), the Harvard Library, and Harvard University Information Technology (HUIT): the Harvard Dataverse is a storehouse for sharing, referring to, investigating, and safeguarding research information. It is available to all logical information from all disciplines around the world.

The Times of India (additionally known by its shortened form 'TOI') is an Indian English-language day by day paper and computerized news media possessed and overseen by The Times Group. It is the third-biggest paper in India by dissemination and the biggest selling English-language day by day on the planet. It is the most established English-language paper in India, and the second-most established Indian paper still available for use, with its first version distributed in 1838. It is nicknamed as "The Old Lady of Bori Bunder", and is an Indian "paper of record".

CHAPTER-3

MODULE DESCRIPTION

3.1 Numeric Analysis

❖ VISUALIZING OPENING, CLOSING PRICES AND STOCK RETURNS

❖ TIME SERIES ANALYSIS

- **DICKEY-FULLER TEST:**The Dickey–Fuller test tests the null hypothesis that a unit root is present in an autoregressive time series model. The alternative hypothesis is different depending on which version of the test is used, but is usually stationarity or trend-stationarity.
- **ROLLING STATISTICS:**A rolling analysis of a time series model is often used to assess the model's stability over time. When analyzing financial time series data using a statistical model, a key assumption is that the parameters of the model are constant over time. However, the economic environment often changes considerably, and it may not be reasonable to assume that a model's parameters are constant. A common technique to assess the constancy of a model's parameters is to compute parameter estimates over a rolling window of a fixed size through the sample. If the parameters are truly constant over the entire sample, then the estimates over the rolling windows should not be too different. If the parameters change at some point during the sample, then the rolling estimates should capture this instability.
- **LOG TRANSFORMATION:**Since the data displayed changing variance over time, the first thing we will do is stabilize the variance by applying log transformation using the `log()` function. The resulting series will be a linear time series. Log transforming the dataset is implied to make the distribution of values more linear and better meet the expectations of this statistical test

- ❖ **ARIMA MODEL:**An autoregressive integrated moving average, or ARIMA, is a statistical analysis model that uses time series data to either better understand the data set or to predict future trends.

A statistical model is autoregressive if it predicts future values based on past values. For example, an ARIMA model might seek to predict a stock's future prices based on its past performance or forecast a company's earnings based on past periods.

An autoregressive integrated moving average model is a form of regression analysis that gauges the strength of one dependent variable relative to other changing variables. The model's goal is to predict future securities or financial market moves by examining the differences between values in the series instead of through actual values.

❖ PREDICTION AND VALIDATION

3.2 Textual Analysis

- ★ SENTIMENTAL ANALYSIS: Sentiment analysis is the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information.

- ★ CATEGORIZING BASED ON CITY AND TYPE OF NEWS

- ★ ASSIGNING POLARITY
 - VADER SENTIMENT INTENSITY ANALYZER: VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. VADER uses a combination of A sentiment lexicon is a list of lexical features (e.g., words) which are generally labeled according to their semantic orientation as either positive or negative. VADER not only tells about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is.

- ★ MSE AND RMSE ON TRAINING AND TESTING
 - DECISION TREE
 - RANDOM FOREST
 - XG BOOST
 - LG BOOST
 - ADA BOOST

Algorithms

1) Decision Tree

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

2) Random Forest

- Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.
- As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.
- The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

3) XG Boost

- XGBoost stands for eXtreme Gradient Boosting.
- XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.
- XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements Machine Learning algorithms under the Gradient Boosting framework. It provides a parallel tree boosting to solve many data science problems in a fast and accurate way.

4) LG Boost

- Light Gradient is a gradient boosting framework based on decision trees to increase the efficiency of the model and reduce memory usage.
- It uses two novel techniques: Gradient-based One Side Sampling and Exclusive Feature Bundling (EFB) which fulfills the limitations of histogram-based algorithm that is primarily used in all GBDT (Gradient Boosting Decision Tree) frameworks. The two techniques of GOSS and EFB comprise together to make the model work efficiently and provide it a cutting edge over other GBDT frameworks

5) ADA Boost

- AdaBoost algorithm, short for Adaptive Boosting, is a Boosting technique used as an Ensemble Method in Machine Learning.
- It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights assigned to incorrectly classified instances.
- Boosting is used to reduce bias as well as variance for supervised learning. It works on the principle of learners growing sequentially.
- Except for the first, each subsequent learner is grown from previously grown learners. In simple words, weak learners are converted into strong ones.

CHAPTER-4

RESULTS AND DISCUSSIONS

4.1 Numeric Analysis

```
stocks.head()
```

	High	Low	Open	Close	Volume	Adj Close
Date						
2009-12-31	7.619643	7.520000	7.611786	7.526071	352410800.0	6.452591
2010-01-04	7.660714	7.585000	7.622500	7.643214	493729600.0	6.553026
2010-01-05	7.699643	7.616071	7.664286	7.656429	601904800.0	6.564356
2010-01-06	7.686786	7.526786	7.656429	7.534643	552160000.0	6.459940
2010-01-07	7.571429	7.466071	7.562500	7.520714	477131200.0	6.447997

```
stocks.reset_index(inplace=True)  
stocks.tail()
```

	Date	High	Low	Open	Close	Volume	Adj Close
3002	2021-12-03	164.960007	159.720001	164.020004	161.839996	117938300.0	161.839996
3003	2021-12-06	167.880005	164.279999	164.289993	165.320007	107497000.0	165.320007
3004	2021-12-07	171.580002	168.339996	169.080002	171.179993	120405400.0	171.179993
3005	2021-12-08	175.960007	170.699997	172.130005	175.080002	116998900.0	175.080002
3006	2021-12-09	176.750000	173.919998	174.910004	174.559998	107976382.0	174.559998

FIG1: Stock values of initial and final days of consideration

```
stocks.describe()
```

	High	Low	Open	Close	Volume	Adj Close
count	3007.000000	3007.000000	3007.000000	3007.000000	3.007000e+03	3007.000000
mean	42.457363	41.592557	42.021811	42.043630	2.711067e+08	40.304090
std	37.389221	36.543409	36.955473	36.990429	2.261654e+08	37.491676
min	7.000000	6.794643	6.870357	6.858929	4.100000e+07	5.880606
25%	18.622321	18.258036	18.477500	18.487679	1.098044e+08	16.114889
50%	28.450001	27.900000	28.167500	28.190001	1.854888e+08	26.013899
75%	48.127501	47.503750	47.878750	47.821251	3.720108e+08	46.243723
max	176.750000	173.919998	174.910004	175.080002	1.880998e+09	175.080002

FIG2: Statistical overview of stock data

```
stocks.isna().sum()
```

```
Date      0  
High      0  
Low       0  
Open      0  
Close     0  
Volume    0  
Adj Close 0  
dtype: int64
```

FIG3: No missing values which implements it's a clean data



FIG4: Analyzing close price



FIG5: Analyzing open price

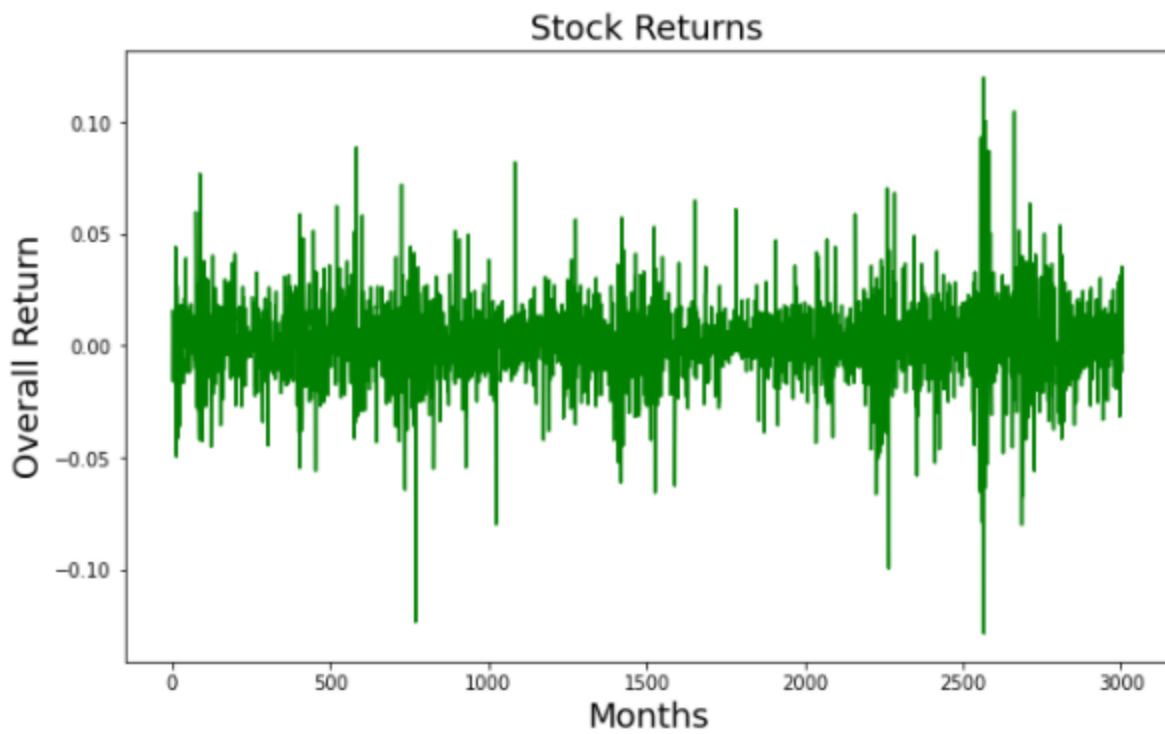


FIG6: Overall Stock Return

Mean and Standard Deviation on transformed data

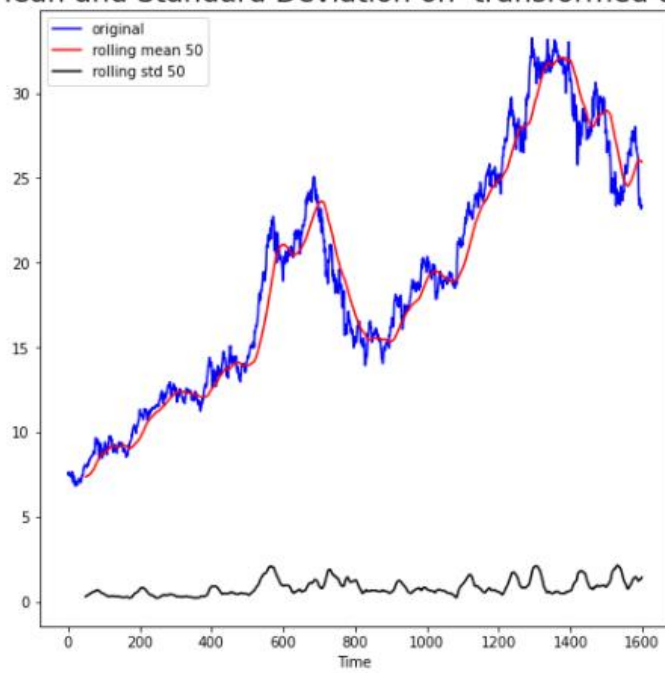


FIG 7: Rolling statistics with fluctuating Variance

Mean and Standard Deviation on Log transformed data

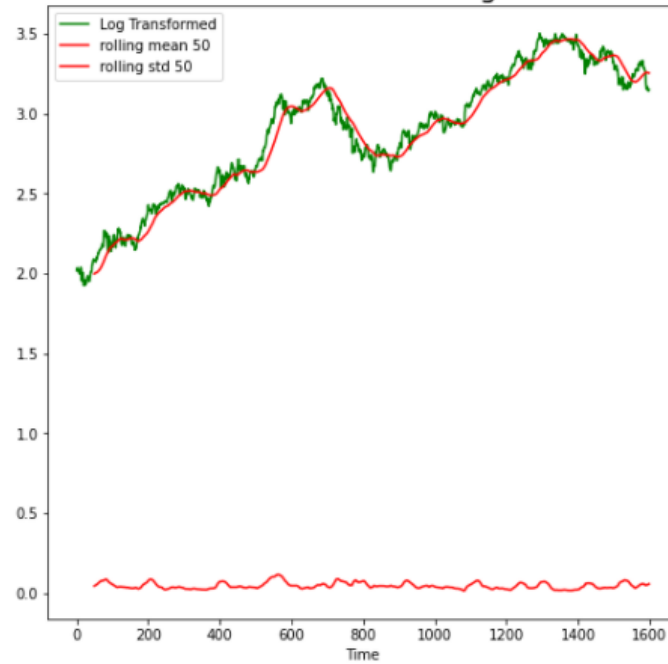


FIG 8: Rolling statistics after log transformation

Mean and Standard Deviation on Differential Log Transformed data

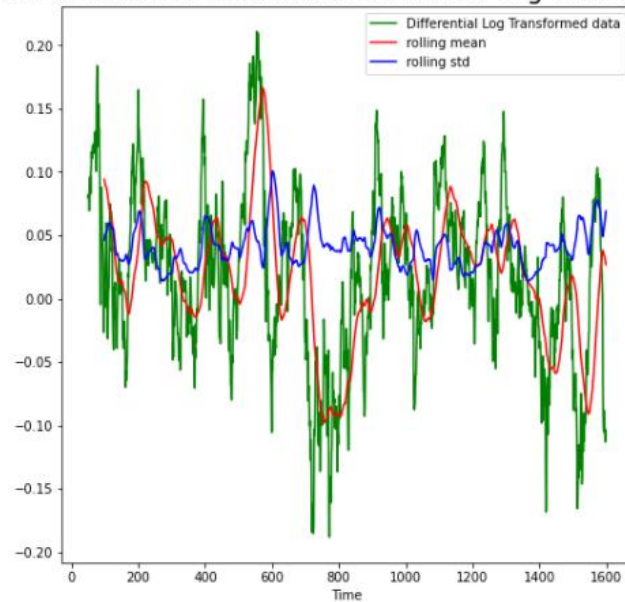


FIG9: Stock returns after linearising

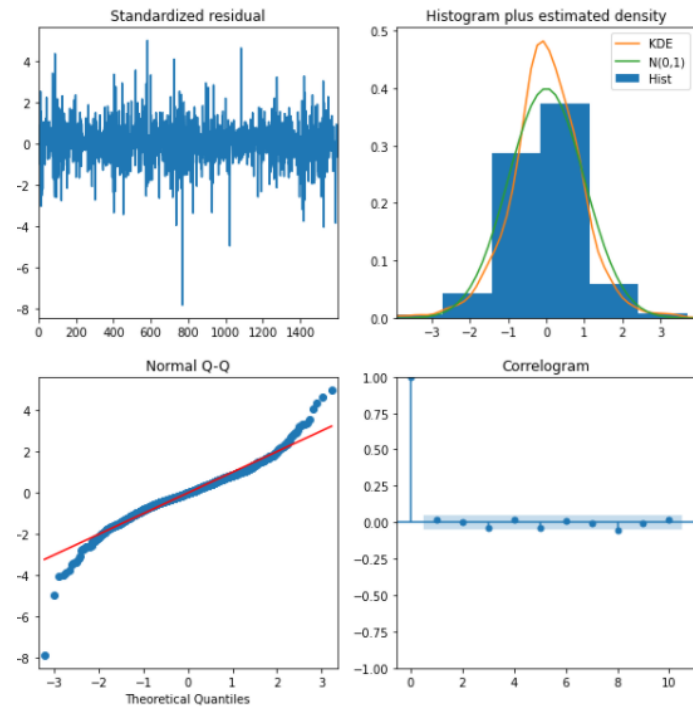


FIG 10: Plots of ARIMA Model

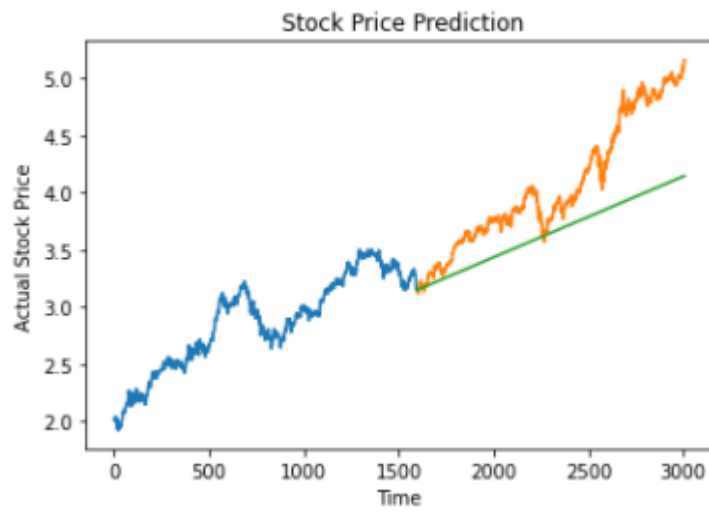


FIG 10: Numeric stock price prediction

```
print('Mean Squared Error      ',mean_squared_error(test_log, predict_ClosePrice))
print('Root Mean Squared Error ',np.sqrt(mean_squared_error(test_log, predict_ClosePrice)))
print('Mean Absolute Error      ',mean_absolute_error(test_log, predict_ClosePrice))
print('R-Squared                ',r2_score(test_log, predict_ClosePrice))
```

```
Mean Squared Error      0.25227254332558063
Root Mean Squared Error 0.5022674022127861
Mean Absolute Error      0.41473918081782324
R-Squared                0.16515127976260247
```

FIG 11: Validating Accuracy of Model

4.2 Textual Analysis

```
news=pd.read_csv("news.csv")
news.head()
```

	publish_date	headline_category	headline_text
0	20010102	unknown	Status quo will not be disturbed at Ayodhya; s...
1	20010102	unknown	Fissures in Hurriyat over Pak visit
2	20010102	unknown	America's unwanted heading for India?
3	20010102	unknown	For bigwigs; it is destination Goa
4	20010102	unknown	Extra buses to clear tourist traffic

FIG 12: Initial News Dataset

```
news.isna().sum()
```

```
publish_date      0
headline_category  0
headline_text      0
dtype: int64
```

FIG 13: Checking Null Values

```
news['headline_category'].value_counts().head()
```

```
india      288541
unknown    209582
city.mumbai 134428
city.delhi  127717
business.india-business 116761
Name: headline_category, dtype: int64
```

FIG 14: Finding Categories and News

```
cities.drop('headline_category', inplace =True,axis =1)
cities.head()
```

	publish_date	headline_text	city_name
273	2001-01-04	Three in race for chief secy's post	bengaluru
274	2001-01-04	Druggists' stir leads to shortage of medicines	patna
277	2001-01-04	He's not so inscrutable	bengaluru
278	2001-01-04	DPCC stages Nyay rally	delhi
642	2001-01-10	Fend for yourselves; Pande tells doctors	patna

FIG 15: Categorizing based on city

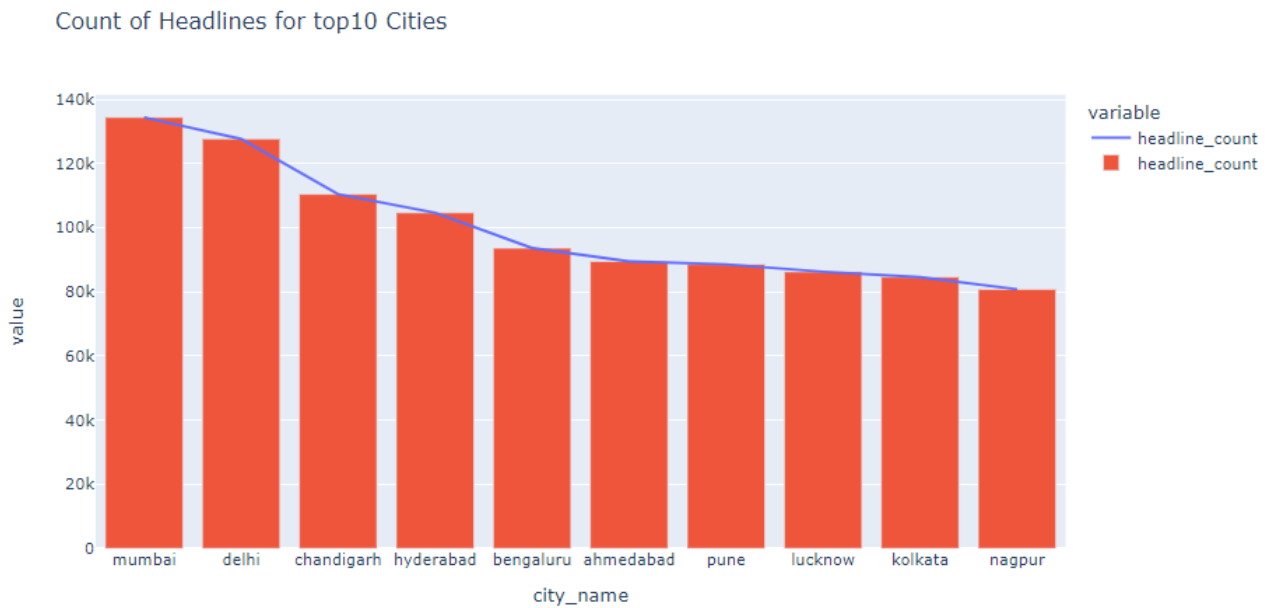


FIG 16: Top 10 Cities with most Headlines

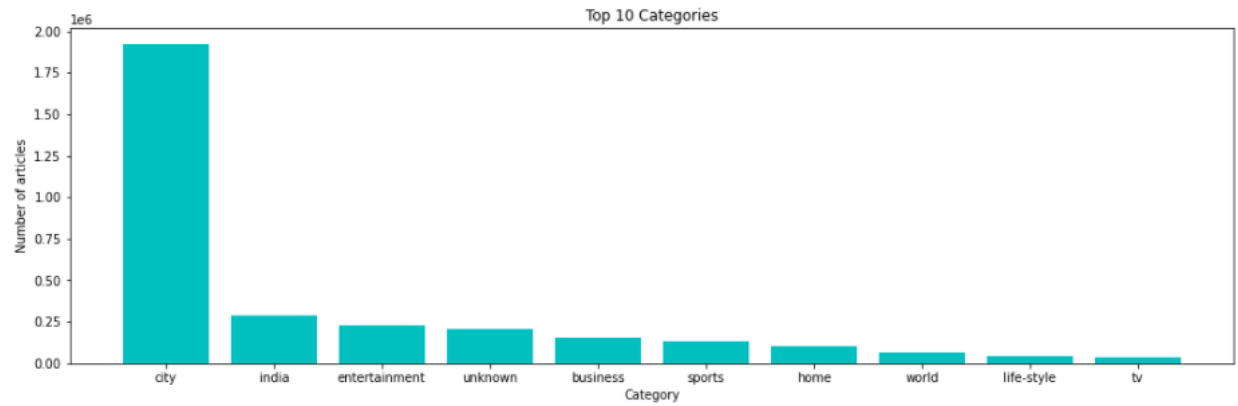


FIG 17: Analyzing based on news Category

```
: news['Compound'] = [senti.polarity_scores(s)['compound'] for s in news['headline_text']]
news['Negative'] = [senti.polarity_scores(s)['neg'] for s in news['headline_text']]
news['Neutral'] = [senti.polarity_scores(s)['neu'] for s in news['headline_text']]
news['Positive'] = [senti.polarity_scores(s)['pos'] for s in news['headline_text']]
```

```
: news.head()
```

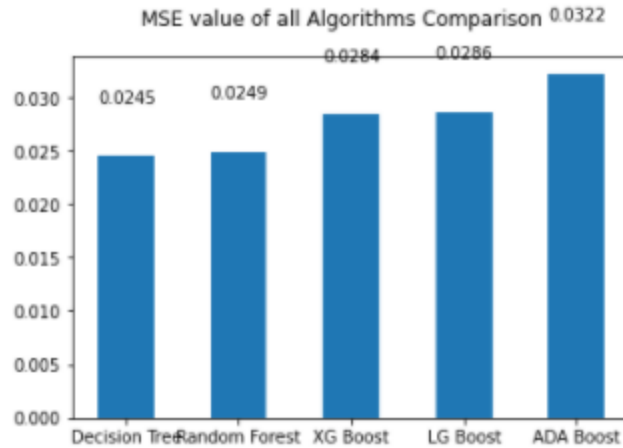
	Date	headline_text	category	Compound	Negative	Neutral	Positive	Subjectivity	Polarity
0	2001-01-02	Status quo will not be disturbed at Ayodhya; s...	unknown	0.2924	0.000	0.805	0.195	0.000000	0.00
1	2001-01-02	Fissures in Hurriyat over Pak visit	unknown	0.0000	0.000	1.000	0.000	0.000000	0.00
2	2001-01-02	America's unwanted heading for India?	unknown	-0.2263	0.322	0.678	0.000	0.000000	0.00
3	2001-01-02	For bigwigs, it is destination Goa	unknown	0.0000	0.000	1.000	0.000	0.000000	0.00
4	2001-01-02	Extra buses to clear tourist traffic	unknown	0.3818	0.000	0.658	0.342	0.241667	0.05

FIG 18: Assigning sentiment to news

	Date	High	Low	Open	Close	Volume	Adj Close	headline_text	category	Compound	Negative	Neutral	Positive	Subjectivity	Polarity
0	2009-12-31	7.619643	7.52	7.611786	7.526071	352410800.0	6.452592	Why aging prevents sleep from enhancing memory	life-style	0.0772	0.000	0.822	0.178	0.0	0.000
1	2009-12-31	7.619643	7.52	7.611786	7.526071	352410800.0	6.452592	Abhishek will die in Delhi-6!	entertainment	-0.6360	0.512	0.488	0.000	0.0	0.000
2	2009-12-31	7.619643	7.52	7.611786	7.526071	352410800.0	6.452592	Don't typecast me: Omi	entertainment	0.0000	0.000	1.000	0.000	0.0	0.000
3	2009-12-31	7.619643	7.52	7.611786	7.526071	352410800.0	6.452592	Tonight's the night and 2010: the year!	entertainment	0.0000	0.000	1.000	0.000	0.0	0.000
4	2009-12-31	7.619643	7.52	7.611786	7.526071	352410800.0	6.452592	Vir's sexy sense of humour!	entertainment	0.7777	0.000	0.306	0.694	1.0	0.625

FIG 19: Combining stock data with news and it's polarity

Metrics calculated while TRAINING the model
 For Decision Tree MSE-Value is 0.0245
 For Random Forest MSE-Value is 0.0249
 For XG Boost MSE-Value is 0.0284
 For LG Boost MSE-Value is 0.0286
 For ADA Boost MSE-Value is 0.0322



Evaluating the model on TESTING DATA
 For Decision Tree MSE-Value is 0.0322
 For Random Forest MSE-Value is 0.0294
 For XG Boost MSE-Value is 0.0286
 For LG Boost MSE-Value is 0.0286
 For ADA Boost MSE-Value is 0.0322

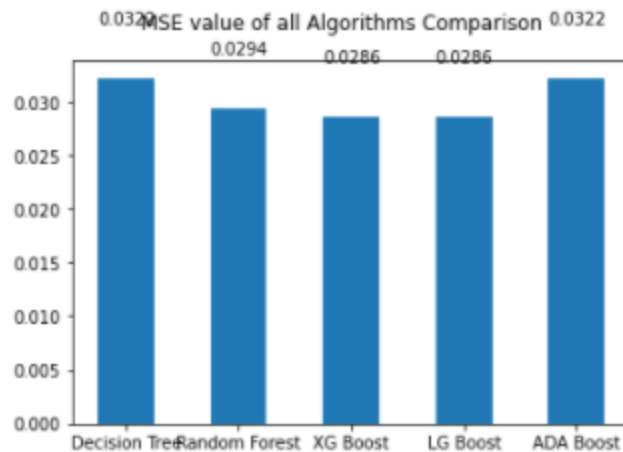
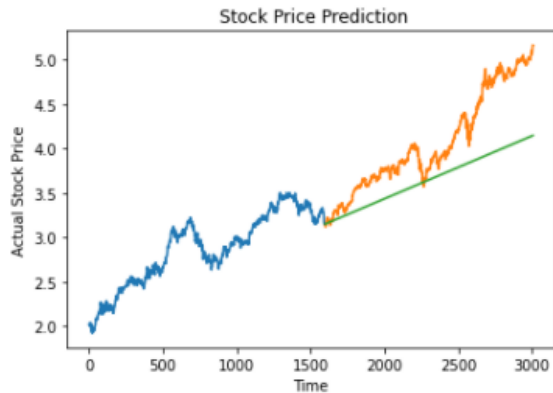


FIG 20: Analysing the textual data to predict which algorithm suits best and be accurate between training and testing data

CHAPTER-5

CONCLUSION

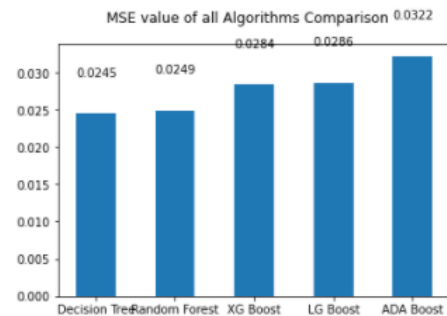


Numeric Prediction

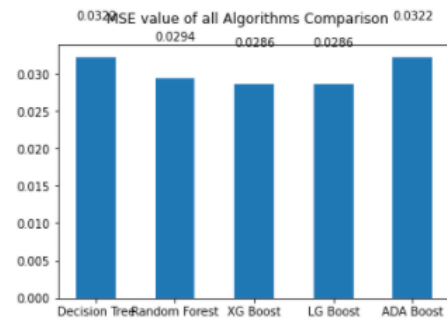
As we can clearly see from the graph, we have predicted the stock market values over a very long period of time (10 years) using training and testing dataset of 10 years each.

And this was achieved with a very low RMS value of 0.25 which itself stands out as a big positive of this project.

Metrics calculated while TRAINING the model
For Decision Tree MSE-Value is 0.0245
For Random Forest MSE-Value is 0.0249
For XG Boost MSE-Value is 0.0284
For LG Boost MSE-Value is 0.0286
For ADA Boost MSE-Value is 0.0322



Evaluating the model on TESTING DATA
For Decision Tree MSE-Value is 0.0322
For Random Forest MSE-Value is 0.0294
For XG Boost MSE-Value is 0.0286
For LG Boost MSE-Value is 0.0286
For ADA Boost MSE-Value is 0.0322



Textual Analysis

After categorizing each and every headline based on their city and type of news, we were able to train and test them extensively.

And later, we are analyzing these trained and tested data using 5 different algorithms as described earlier. This is compared and analyzed based on RMS values.

As, we know that the model which performs closely accurate in training and testing data, and also has a rms value close to zero is the best model.

So, according to those 2 criterias, we were able to conclude that LG Boost is the best regressor model to be used in this kind of textual analysis.

REFERENCE

- ❖ Sharma, Ashish, Dinesh Bhuriya, and Upendra Singh. "Survey of stock market prediction using machine learning approach." 2017 international conference of electronics, communication and aerospace technology (ICECA). Vol. 2. IEEE, 2017.
- ❖ Gandhmal, Dattatray P., and K. Kumar. "Systematic analysis and review of stock market prediction techniques." Computer Science Review 34 (2019): 100190.
- ❖ Yoo, Paul D., Maria H. Kim, and Tony Jan. "Machine learning techniques and use of event information for stock market prediction: A survey and evaluation." International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06). Vol. 2. IEEE, 2005.
- ❖ Li, Xiaodong, Pangjing Wu, and Wenpeng Wang. "Incorporating stock prices and news sentiments for stock market prediction: A case of Hong Kong." Information Processing & Management 57.5 (2020): 102212.
- ❖ Li X, Wu P, Wang W. Incorporating stock prices and news sentiments for stock market prediction: A case of Hong Kong. Information Processing & Management. 2020 Sep 1;57(5):102212.

APPENDIX

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import pylab as pl

%matplotlib inline

import seaborn as sns

import datetime

import pandas_datareader.data as web

import matplotlib as mpl

from matplotlib import style

import warnings

warnings.simplefilter(action='ignore', category=FutureWarning)

warnings.filterwarnings('ignore')

from statsmodels.tsa.stattools import adfuller, acf, pacf

from statsmodels.tsa.arima_model import ARIMA

from sklearn.metrics import mean_squared_error,
mean_absolute_error, r2_score

from statsmodels.tsa.seasonal import seasonal_decompose

from textblob import TextBlob

from matplotlib.pyplot import figure

from matplotlib import rcParams

import plotly.express as px
```

```
#from wordcloud import WordCloud, STOPWORDS

from nltk.sentiment.vader import SentimentIntensityAnalyzer

import nltk

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor

import xgboost

import lightgbm

stocks.head()

stocks.reset_index(inplace=True)

stocks.tail()

stocks.shape

stocks.describe()

stocks['Date'] = pd.to_datetime(stocks['Date'])

stocks.head()

stocks.isna().sum()

plt.figure(figsize=(5,5))

plt.title('Closing Price of Stocks', fontsize = 18)

plt.xlabel('Days', fontsize= 18)

plt.ylabel('Close', fontsize = 18)

plt.plot(stocks['Close'])

plt.show()

plt.figure(figsize=(5,5))

plt.grid(True)
```

```

plt.plot(stocks['Open'])

plt.xlabel('Days')

plt.ylabel('Open Price')

plt.title('Opening price of Stocks')

plt.show()

close = stocks['Close']

returns = close / close.shift(1) - 1

plt.figure(figsize = (10,6))

returns.plot(label='Return', color = 'g')

plt.title("Stock Returns")

train = stocks[:1600]

test = stocks[1600:]

train.shape, test.shape

def adfullerTest(X):

    result = adfuller(X,autolag = 'AIC')

    print('ADF Statistic: %f' % result[0])

    print('p-value: %f' % result[1])

    print('No of Lags Used: %f' % result[2])

    print('Number of Obs Used: %f' % result[3])

    print('Critical Values:')

    for key, value in result[4].items():

        print('\t%s: %.3f' % (key, value))

    if result[1] <=0.05 :

```

```

        print("Reject against the null hypothesis, time series is
stationary")

    else:

        print("Accept null hypothesis, time series is non-stationary ")

adfullerTest(train['Close'])

rolling_mean_50 = (train['Close']).rolling(window=50).mean()

rolling_std_50 = (train['Close']).rolling(window=50).std()

plt.figure(figsize = (8,8))

plt.plot((train['Close']), color = 'blue', label = 'original')

plt.plot(rolling_mean_50, color = 'red', label = 'rolling mean 50')

plt.plot(rolling_std_50, color = 'black', label = 'rolling std 50')

plt.xlabel('Time')

plt.legend()

plt.title('Mean and Standard Deviation on  transformed data',fontsize=20)


train_log = np.log(train['Close'])

test_log = np.log(test['Close'])

from numpy import log

adfullerTest(log(train['Close']))

rolling_mean_50 = log(train['Close']).rolling(window=50).mean()

rolling_std_50 = log(train['Close']).rolling(window=50).std()

plt.figure(figsize = (8,8))

plt.plot(log(train['Close']), color = 'g', label = 'Log Transformed')

plt.plot(rolling_mean_50, color = 'r', label = 'rolling mean 50')

```

```

plt.plot(rolling_std_50, color = 'r', label = 'rolling std 50')

plt.xlabel('Time')

plt.legend()

plt.title('Mean and Standard Deviation on Log transformed data', fontsize
= 20)

mean_log = log(train['Close']).rolling(50).mean()

train_log_diff = log(train['Close']) - mean_log

train_log_diff.dropna(inplace = True)

adfullerTest(train_log_diff)

data= train_log_diff

mean = data.rolling(50).mean()

std = data.rolling(50).std()

plt.figure(figsize = (8,8))

plt.plot(data, color = 'g', label = 'Differential Log Transformed data')

plt.plot(mean, color = 'r', label = 'rolling mean')

plt.plot(std, color = 'b', label = 'rolling std')

plt.xlabel('Time')

plt.legend()

plt.title('Mean and Standard Deviation on Differential Log Transformed
data', fontsize = 20)

import pmdarima as pmd

def arimamodel(timeseriesarray):

    autoarima_model = pmd.auto_arima(timeseriesarray,

```

```

        # start_p=1,

        #start_q=1,

        #test="adf",

        trace=True,

        error_action = 'ignore',

        suppress_warnings = True)

    return autoarima_model

stocks_arima = arimamodel((train_log))

stocks_arima.summary()

stocks_arima.plot_diagnostics(figsize=(10,10))

plt.show()

predict_ClosePrice = stocks_arima.predict(n_periods = len(test_log))

predict_ClosePrice = pd.DataFrame(predict_ClosePrice,index =
test_log.index,columns=['predict_ClosePrice'])

plt.plot(train_log, label='Train')

plt.plot(test_log, label='Test')

plt.plot(predict_ClosePrice, label='Prediction')

plt.title(' Stock Price Prediction')

plt.xlabel('Time')

plt.ylabel('Actual Stock Price')

print('Mean Squared Error      ',mean_squared_error(test_log,
predict_ClosePrice))

print('Root Mean Squared Error ',np.sqrt(mean_squared_error(test_log,
predict_ClosePrice)))

```

```

print('Mean Absolute Error      ',mean_absolute_error(test_log,
predict_ClosePrice))

print('R-Squared                  ',r2_score(test_log, predict_ClosePrice))

news=pd.read_csv("news.csv")

news.head()

news['publish_date'] = pd.to_datetime(news['publish_date'],format=
'%Y%m%d')

news.head()

news['publish_date'] = pd.to_datetime(news['publish_date'],format=
'%Y%m%d')

news.tail()

news.isna().sum()

news.shape

(news.columns)

news['headline_category'].value_counts().head()

cities = news[news['headline_category'].str.contains('^city\.[a-z]+$',
regex=True)]

cities.head(10)

city = pd.DataFrame(columns = ['city_name'])

city['city_name'] = cities.headline_category.str.split('.',expand =
True)[1]

cities = pd.concat([cities, city], axis = 1)

cities.head()

cities.drop('headline_category', inplace =True,axis =1)

cities.head()

```

```

cities =
cities.groupby(cities['city_name']).agg({'headline_text':'count'})

cities.head()

cities.rename(columns = {'headline_text':'headline_count'}, inplace =
True)

cities = cities.sort_values(by='headline_count',ascending=False)

cities.head()

top10cites = cities.head(10)

def fig_plot(top10cities,title1):

    fig = px.line(top10cities,title =title1)

    for i in top10cities.columns[0:]:

        fig.add_bar(x= top10cities.index ,y =
top10cities['headline_count'],name = i)

    fig.show()

fig_plot(top10cites,'Count of Headlines for top10 Cities')

cities.head()

news.head()

news['category']=news['headline_category'].str.split('.').map(lambda x :
x[0])

categories =
news.groupby(['category']).agg({'headline_text':'count'}).sort_values(by='
headline_text',ascending = False)

news_cat=categories.head(10)

news_cat.reset_index(inplace = True)

news_cat

import matplotlib.colors as mcolors

```



```

plt.figure(figsize=(17,5))

plt.bar(news_cat.category,height= news_cat.headline_text, color = 'c')

plt.xlabel('Category')

plt.ylabel('Number of articles')

plt.title('Top 10 Categories')

plt.show()

news.drop('headline_category', inplace = True, axis =1)

news.head()

# Create a function to get the subjectivity

def Subjectivity(text):

    return TextBlob(text).sentiment.subjectivity


# Create a function to get the polarity

def Polarity(text):

    return TextBlob(text).sentiment.polarity


news['Subjectivity'] =news['headline_text'].apply(Subjectivity)

news['Polarity'] =news['headline_text'].apply(Polarity)

import nltk

from nltk.sentiment.vader import SentimentIntensityAnalyzer as SIA

senti = SIA()

news['Compound'] = [senti.polarity_scores(s)['compound'] for s in
news['headline_text']]

news['Negative'] = [senti.polarity_scores(s)['neg'] for s in
news['headline_text']]

```

```

news['Neutral'] = [senti.polarity_scores(s)['neu'] for s in
news['headline_text']]

news['Positive'] = [senti.polarity_scores(s)['pos'] for s in
news['headline_text']]

news.head()

news.tail()

news.rename(columns = {'publish_date':'Date'}, inplace = True)

df_merge = pd.merge(stocks, news, how='inner', on=['Date'])

df_merge.head()

df = df_merge[['Close', 'Subjectivity', 'Polarity', 'Compound', 'Negative',
'Neutral' , 'Positive']]

df.head()

from sklearn.preprocessing import MinMaxScaler

sc = MinMaxScaler()

new_df = pd.DataFrame(sc.fit_transform(df))

new_df.columns = df.columns

new_df.index = df.index

new_df.head()

X = new_df.drop('Close', axis=1)

y =new_df['Close']

from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.2,
random_state = 0)

X_train.shape , X_test.shape

def func_graph(results,names):

```

```

fig = plt.figure()

fig.suptitle('MSE value of all Algorithms Comparison')

ax = fig.add_subplot(111)

width = 0.5

bars=plt.bar(names,results, width, align='center')

ax.set_xticklabels(names)

for bar in bars:

    yval = bar.get_height()

    plt.text(bar.get_x(), yval +0.005, yval)

plt.show()

from sklearn import metrics

def metric_calc(name,model,category, X_train, Y_train, X_test, Y_test):

    if category =='TRAINING DATA' :

        X_data= X_train

        Y_data=Y_train

    else :

        X_data= X_test

        Y_data=Y_test

    model.fit(X_train, Y_train)

    predictions = model.predict(X_data)

    mse =round(metrics.mean_squared_error(predictions,Y_data),4)

```

```

    print('For ', name, 'MSE-Value is ', mse)

    return mse

def func_modelling(i) :

    count=0

    count=count+1

    X = X_train[i]

    Y = Y_train

    x_test = X_test[i]

    seed = 7

    # preparing models list

    models = []

    models.append(('Decision Tree',' DecisiontreeRegressor ',
DecisionTreeRegressor()))

    models.append(('Random Forest',' RandomForestRegressor ',
RandomForestRegressor()))

    models.append(('XG Boost',' XGBRegressor ', xgboost.XGBRegressor()))

    models.append(('LG Boost',' LGBMRegressor ',
lightgbm.LGBMRegressor()))

    models.append(('ADA Boost',' AdaBoostRegressor ',
AdaBoostRegressor()))

    results_train = []

    results_test = []

    names = []

    scoring = 'MSE'

    print('Metrics calculated while TRAINING the model')

```

```
for name,label, model in models:

    cv_results_train=metric_calc(name,model, 'TRAINING DATA',X,Y,
x_test,Y_test)

    results_train.append(cv_results_train)

    names.append(name)

func_graph(results_train,names)


print('Evaluating the model on TESTING DATA')

for name,label, model in models:

    cv_results_test=metric_calc(name,model, 'TESTING DATA',X,Y,
x_test,Y_test)

    results_test.append(cv_results_test)

    #names.append(name)

func_graph(results_test,names)

func_modelling(X_train.columns)
```